Modeling Audio Distortion Effects with Autoencoder Neural Networks

Riccardo Russo(✉), Francesco Bigoni, and George Palamas

Aalborg University Copenhagen, Copenhagen, Denmark
{rrusso19,fbigon17}@student.aau.dk, gpa@create.aau.dk

Abstract. Most music production nowadays is carried out using software tools: for this reason, the market demands faithful audio effect simulations. Traditional methods for modeling nonlinear systems are effect-specific or labor-intensive; however, recent works yielded promising results by black-box simulation of these effects using neural networks. This work aims to explore two models of distortion effects based on autoencoders: one makes use of fully-connected layers only, and the other employs convolutional layers. Both models were trained using clean sounds as input and distorted sounds as target, thus, the learning method was not self-supervised, as it is mostly the case when dealing with autoencoders. The networks were then tested with visual inspection of the output spectrograms, as well as with an informal listening test, and performed well in reconstructing the distorted signal spectra, however a fair amount of noise was also introduced.

Keywords: Autoencoders · Convolutional autoencoders · Audio distortion · Audio effects modeling · Black box modeling · Machine learning for audio

1 Introduction

Through audio effects, one can manipulate an audio signal in order to shape its characteristics to fit the desired purpose. These tools have important creative and industrial applications in areas such as music production or telecommunications. In the music field, audio effects are generally divided into macro-categories which either refer to the audio feature being transformed or to the method used, e.g. *dynamic range compressor, distortion, pitch shifter, phase vocoder*, etc. Among all, distortion effects aim to increase the amplitude and enrich the spectral content of the input by making use of nonlinear components. In the digital domain, this kind of transformations entails the use of nonlinear functions to process the incoming waveform into a different shape, which depends on the amplitude of the incoming signal.

While audio processing devices were originally based on analog circuits, most music production nowadays is carried out using digital audio workstations

N. Shaghaghi et al. (Eds.): INTETAIN 2020, LNICST 377, pp. 131–141, 2021.
https://doi.org/10.1007/978-3-030-76426-5_9

(DAWs) and audio plugins: for this reason, the demand for accurate digital emulations of analog audio effects is always high. The field of virtual analog (VA) modeling is concerned with creating these emulations, and over time many commercial solutions have been proposed, such as AmpliTube from IK Multimedia[1] or Guitar Rig from Native Instruments[2]. A common approach for analog modeling is the so-called *white-box modeling* [3]. This technique involves analysing the circuitry of the device and simulating it through discrete-time mathematical models. White-box modeling is a widely used method and can be very accurate, but has some drawbacks. First, it requires a exhaustive knowledge of the circuit under exam, which is not always possible. Second, the simulations can be computationally demanding, especially if the circuit contains many nonlinear components. Lastly, this technique requires intensive labor for the design of a single effect.

An alternative approach is *black-box modeling* [3], which is based on measuring the response of the system to particular input signals, in order to create an input-output map. Since this technique is based on measurements of the device under test, it is easier to adapt it for the simulation of different effects. However, black-box modeling comes with drawbacks too, e.g. if user controls are required, one has to measure the system response for every parameter configuration. A nonlinear system cannot be modeled using classic frequency response analysis, since this assumes linear and time-invariant systems; for this reason, different mathematical methods have been proposed for addressing this task over time [14]. More recently, deep learning [5] methods have been explored, showing increasingly good results [2–4,8–10,16]. This paper is organized as follows: Sect. 2 explores two techniques for audio distortion modeling, both based on convolution. In Sect. 3 two models of distortion effects are presented, and the results are discussed in Sect. 4. Conclusions are drawn in Sect. 5.

2 Background

As stated above, neural networks have found many applications in audio signal processing. However, common practice has been to not work directly in the time domain, but rather use time-frequency representations (e.g. spectrograms) as input [7]. Despite its many applications, this approach involves discarding the phase information, which is an important feature of audio distortion. The nonlinear phase behaviour of many audio effects, including distortion, causes the partial cancellation of some frequencies, affecting the sound quality in a perceivable manner. For this reason, only models that take raw audio as input were considered in this work. Recurrent neural networks (RNNs) are a common way to approach the task of generating data with a definite temporal structure, and many works on audio effects have been using this technique [2,15,16]; an extensive evaluation of these methods is behind the scope of this paper. Moreover, WaveNet demonstrated that it is possible to achieve significant results by using only convolutional layers [12].

[1] www.ikmultimedia.com/products/amplitube4.
[2] www.native-instruments.com/en/products/komplete/guitar/guitar-rig-5-pro.

2.1 WaveNet

WaveNet is a generative model operating directly on raw audio waveforms. It is *autoregressive*, meaning that the output variable depends linearly on its own previous values. Based on this technique, WaveNet shapes the discrete probability distribution of the next sample given a series of past samples. Both input and the output have the same dimensions, as the model outputs a probability distribution over the current sample x_t using a softmax layer. An entire sequence of samples is produced by sequentially feeding the previously generated samples back into the model. The probability of a waveform $\mathbf{x} = \{x_1, \ldots, x_T\}$ is given by the following relation:

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1}) \ . \tag{1}$$

Looking at Eq. 1, it is clear that the probability of the sample x_t is conditioned on all the past samples x_n. WaveNet has demonstrated state-of-the-art performances in the field of speech generation, and studies has been made to adapt its structure for other purposes; as an example, Damskägg et al. developed a WaveNet-based model for the simulation of tube amplifiers [4] and distortion effects [3]. The network structure is similar to the one proposed by Rethage et al. and used for speech denoising [13]. These tasks differ from the one for which the original WaveNet architecture was developed. While WaveNet aimed at generating a new audio output, these adaptations are made for processing an existing input. For this reason, these works modify the architecture for a regression task, eliminating the softmax output layer: hence, the model is trained to predict the current audio sample, based on a series of past samples and the current one, as shown in Eq. 2:

$$\hat{y}[n, \theta] = f(x[n], \ldots, x[n - N], \theta) \ , \tag{2}$$

where x is the input, N is the receptive field, θ are the network parameters and f is the nonlinear transformation learned by the model.

The model consists of a series of convolutional layers, which take the original waveform as input and apply a linear filter along with a nonlinear activation function. The layers include residual connections between them.

The loss function used is the *normalized mean squared error (NMSE)*: the model parameters are learned by minimizing the squared error between the output signal and the target, divided by the square of the target, as follows:

$$NMSE = \frac{\sum_{n=0}^{\infty} (y[n] - \hat{y}[n, \theta])^2}{\sum_{n=0}^{\infty} y[n]^2} \tag{3}$$

The tube simulation model described in [4] includes a conditioning control. This feature helps to overcome one of the previously mentioned drawbacks of black-box simulation: the difficulty of including user-controllable parameters. Previous studies on this approach, have yielded promising results [3].

2.2 Autoencoders

Autoencoders (AEs) [6] are data-specific lossy compression algorithms. They are used to obtain a compressed representation of an input, which is stored in the so-called *latent space* and are typically employed for dimensionality reduction and data denoising [7], however, they have been also used for more artistic purposes [11]. Although these networks obtained significant results in the field of audio denoising, the applications in the area of nonlinear effects simulation are still not thoroughly explored. In addition, the simplicity of their structure and the smaller training effort compared to WaveNet-like methods makes them easier to adapt for different tasks. AEs consist of a front end, or *encoder*, a latent space representation, and a back end, or *decoder*. The encoder and decoder usually have the same structure (mirrored, as seen in e.g. Fig. 1). AEs are trained in a self-supervised fashion: since the input values are also the target, the network learns to encode data in order to reconstruct it from its latent space representation. These models can make use of different layer types: for example, we talk about *deep AEs* when encoder and decoder contain more than one dense hidden layer, or *convolutional AEs* when they employ convolutional layers.

Recently, Martìnez Ramìrez et al. proposed a convolutional AE for nonlinear audio effect simulation [10], based on their previous model for performing automatic equalization [9]. This model is entirely time-domain based and is divided into three parts: adaptive front end, synthesis back end and latent-space deep neural network (DNN). The front end consists of two convolutional layers, one pooling layer and a residual connection. The first convolutional layer contains 128 one-dimensional filters with a kernel size of 64, and an *absolute value* activation function. The second layer is equal to the first one, but it is locally connected, meaning that it resembles a filter bank structure; it also applies a *softplus* activation function, i.e. a smoothed version of the *rectified linear unit* (ReLU) function, to the input.

The pooling is obtained through a *max-pooling* layer with a window of 16 samples (i.e. the layer returns the maximum value between the 16 analysed). The latent space consists of a locally-connected dense layer of 64 units, and a fully-connected one of 64 units, both followed by a *softplus* function. The back end is made of an unpooling layer, a DNN with nonlinear activation functions, and a convolutional layer which is exactly the same as the input layer of the front end. The loss function used is the NMSE, same as Eq. 3.

3 Design and Implementation

We developed two models in order to test the performances of different types of AEs: firstly, we implemented a deep AE, using only dense layers; secondly, we replaced some of the dense layers with convolutional layers in a convolutional AE. Both models were implemented using the Keras library[3].

[3] www.keras.io.

3.1 Deep Autoencoder

The deep AE is built with dense layers only, as shown in Fig. 1. An overview of the architecture is presented here:

- The encoder contains 1 input layer (579 units) and three hidden layers (256, 128 and 64 units respectively),
- The latent space is made of a single layer (32 units),
- Finally, the decoder has the same structure as the encoder, but mirrored, so that the output has the same dimensions as the input.

The optimal number of hidden layers was determined empirically by considering different combinations of sizes and number of layers. Each layer applies a ReLU activation function, except for the last layer of the decoder, which has a linear activation function. This approach was chosen rather than re-normalizing the output samples between 0 and 1, as our initial tests showed that it would produce an output amplitude range similar to the input range. In addition, our experiments showed that the data squeezing performed by the ReLU resulted in amplified background noise, making the musical structure in the output sound almost indistinguishable to the ear; by applying a linear function to the last layer instead, we obtained significantly better results.

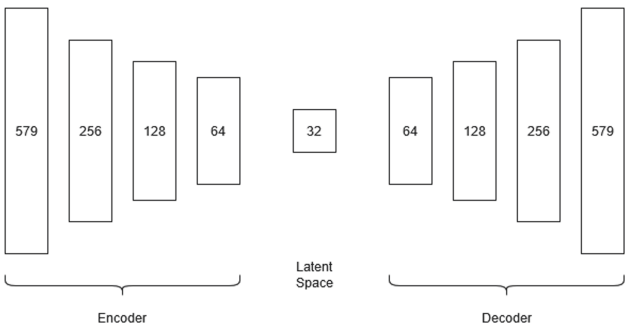


Fig. 1. Structure of the deep autoencoder. The numbers indicate the quantity of fully connected units.

3.2 Convolutional Autoencoder

The convolutional AE was built by taking the deep AE and substituting some dense layers with convolutional layers. Our architecture is depicted in Fig. 2:

- The encoder is made of 1 dense input layer (256 units) and a convolutional layer (128 one-dimensional filters of size 64),
- The data are then flattened to fit a latent space made of a single dense layer (512 units),

– The decoder contains 1 convolutional layer (same hyperparameters as the one in the encoder),
– Lastly, the data are flattened to fit the output dense layer (256 units).

This structure was chosen in order to re-implement some parts of the model proposed by Ramìrez et al. [10]. At a first stage, a max-pooling and an unpooling layer were included, but we chose to remove them since they prevented the network from learning. As for the deep AE, all layers except the last one apply a ReLU activation function to the data (as opposed to Ramìrez et al., that makes use of *softplus* functions). This change was necessary, as testing using softplus yielded far more noisy results than with ReLU.

As for the deep AE model, the last layer of the convolutional AE applies a linear function, for reasons described in the previous subsection.

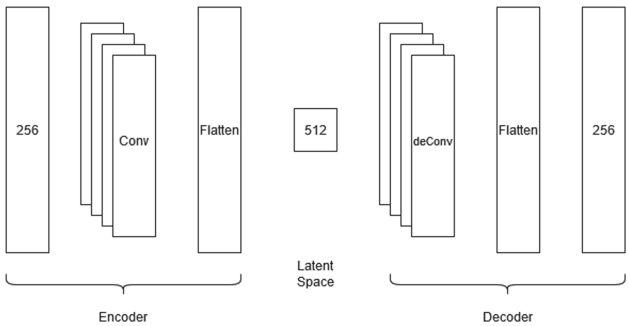


Fig. 2. Structure of the convolutional autoencoder.

3.3 Dataset

Our input data is obtained from the *IDMT-SMT-Audio-Effects* dataset [1], which is made of 2-second long single tones and two-note bichords recorded from various 6-string electric guitars and 4-string bass guitars, and covers the common pitch range of these instruments. The recordings include the clean notes and their respective effected versions. The dataset includes eleven different effects with different settings for each effect: more specifically, three different settings for each tone are present for the distortion effect. Since including user controls falls outside the scope of this paper, only the instances for the first of the three settings were used for training. The training procedure differs from self-supervised learning, which is more commonly used when dealing with autoencoders: unprocessed (clean) monophonic notes were given as input, and the corresponding processed (distorted) notes were set as target. Thus, the network learned to apply the distortion effect to the clean sound. A part of the monophonic set was used for validation, and the trained model was tested on single notes, as well as two-note bichords. The files in the dataset were recorded at a sample rate 44100 Hz (16-bit, mono).

3.4 Training

Figure 3 shows a diagram of the training process. The models were trained using clean sounds as inputs and distorted sounds from the dataset (same pitch as the input) as targets. Both models were trained using the Adam optimizer and a normalized mean squared error (NMSE) loss function was used to compare the processed and target output (Eq. 3). The training was performed on a workstation with three Nvidia TITAN X GPUs and an Intel Xeon CPU.

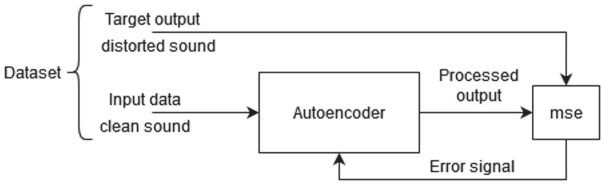


Fig. 3. Diagram of the training process: the processed output is compared to the target distorted sound using a normalized mean squared error loss function.

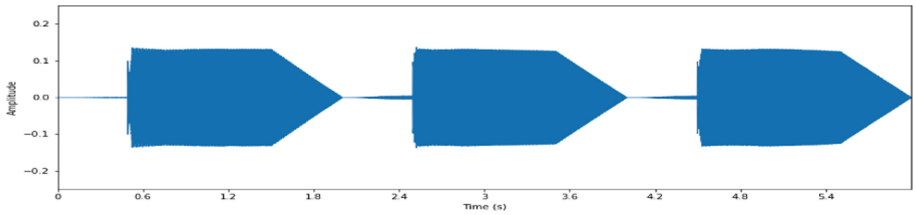
Deep AE. We trained on 13 min of audio, which corresponds to the length of all the distorted notes data with the chosen setting. The sample rate 44100 Hz and a mini-batch size of 128 was used. The audio was cut in slices of 579 samples each (i.e. 13.1 ms) in order to feed the input layer of the network.

Convolutional AE. Only a smaller part of the entire dataset could be used before running out of memory: the training was performed on 3 min of audio, with a mini-batch size of 64. The input data was divided in frames of 256 samples (i.e. 5.8 ms), in order to feed the input layer of the network.

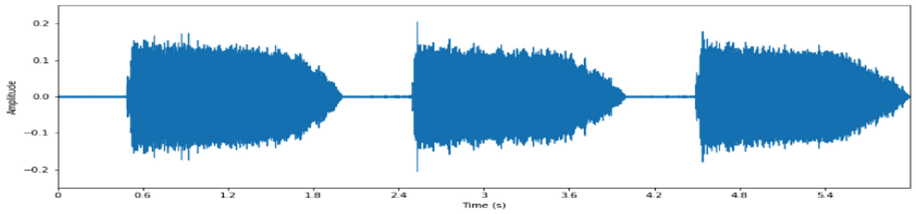
4 Results and Analysis

After the model is trained, a clean input is fed into the AE, which returns a processed (i.e. distorted) output.

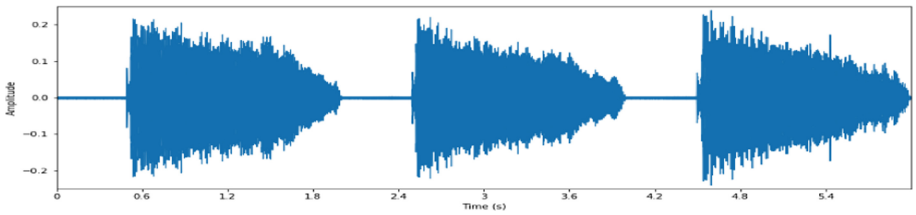
We tested the models with a sequence of clean single notes and two-note bichords. The reconstructed audio was compared to the corresponding target from the dataset through informal listening tests and visual inspection of the spectrograms. As an example, we compare time-domain plots (Fig. 4) and spectrograms (Fig. 5) of three single notes as output of our two architectures vs the target signal from the dataset. In this case, we choose single notes rather than bichords as they allow an easier visual inspection. As the outputs of both models contained broadband noise, we applied lowpass filters with a cutoff frequency of 10 kHz in order to cancel the spurious high frequencies outside the guitar/bass



(a) Target (distorted) guitar sound from the dataset.



(b) Output of the deep AE.



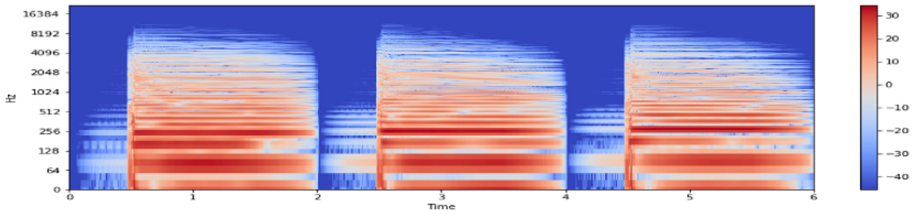
(c) Output of the conv AE.

Fig. 4. Time-domain comparison between target (i.e. distorted sound from the dataset) and the outputs of the deep AE and convolutional AE. Here, three 2-second notes (E2, F2, F#2) played with an electric guitar are shown.

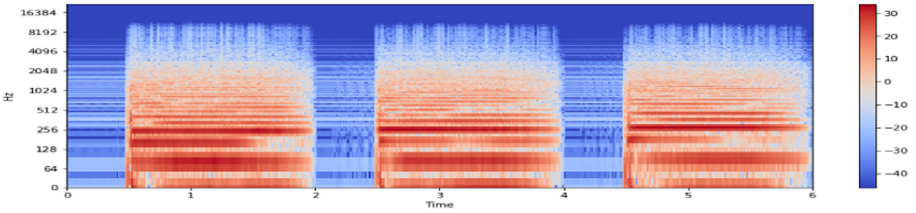
range. The time-domain plots (Fig. 4) show that the model outputs present a rougher shape compared to the target sound, with clear ripples in the case of the convolutional AE. The peak amplitude is larger (again, especially the output of the convolutional AE): this is probably due to the additional noise which can also be heard in the processed sound.

4.1 Deep AE

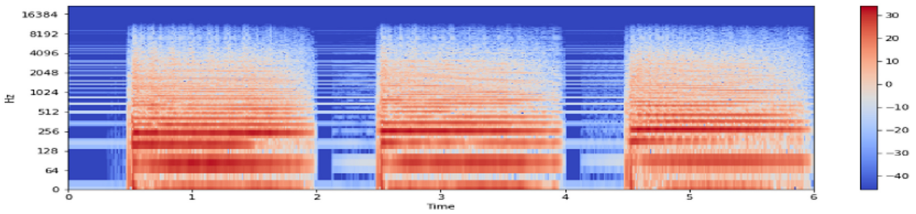
Figure 5b shows the spectrogram of one output from the deep AE. By comparing with the spectrogram of the target sound in Fig. 5a, it can be noted that the reconstructed data are noisy, but the desired spectral content is present almost in its entirety. However, conversely to the target signal, some undesired frequency content is added between the tones: our listening tests suggest that this corresponds to the guitar pickup hum, which is amplified by the distortion effect. In the real world, this sound is usually less intense that the guitar sound and disappears quickly after the attack. This difference may be due to the fact that



(a) Target (distorted) guitar sound from the dataset.



(b) Output of the deep AE.



(c) Output of the convolutional AE.

Fig. 5. Spectrograms of the signals represented in Fig. 4. It should be noted how the most relevant frequency content is reliably reconstructed, but a fair amount of noise is also introduced by the networks.

the network has not been properly trained to generate silence, as it hasn't been trained on instances that contained silence: a further analysis is left for future work. The comparison between waveforms (Figs. 4b and 4a) shows that, despite the aforementioned discrepancies (which contribute in hiding the sharp decay clearly visible in the target signal), the network seems able to approximate the correct amplitude envelope.

4.2 Convolutional AE

Figure 5c shows the spectrogram of the output from our convolutional AE. The spectral reconstruction does not seem to be improved with respect to the deep AE, and our output results even noisier than in the previous case. However, this model performs better while reconstructing the frequency content of the pause between two notes, i.e. the "silent" parts of the signal mentioned above. An inspection of the waveforms (Figs. 4c and 4a) shows that the convolutional AE does a worse amplitude envelope reconstruction than the deep AE: the first

peak is smaller than in the target sound, whereas the main peak is higher, and the envelope exhibits clear ripples. This might be explained with the fact that this model was trained on a less amount of data that the previous one: a more thorough investigation is left for future work.

5 Conclusions and Future Work

In this paper, we gave a background on previously implemented methods for black-box modeling of nonlinear audio effects. We have implemented two models with different hyperparameters: a deep autoencoder and a convolutional autoencoder. These were then trained in a different way with respect to the standard self-supervised manner: clean sounds were used as input and distorted sounds were used as target. Our results show that, despite the noise in the output, even simple architectures as deep autoencoders are valid implementations for the required task. Despite the added complexity, our convolutional autoencoder did not achieve noticeable improvements in terms of either reconstructed spectral content or noise amount when compared to the deep autoencoder: this suggests that a proper deep learning virtual analog model based on convolutional layers requires a more complex structure than the one we implemented, and perhaps a larger training set. However, a more extensive comparison on these results should be performed in order to test the performance of both models properly. As far as noise is concerned, an additional deep autoencoder could be trained on the task of removing it, using the distorted sound from the dataset as target. A further analysis of the number and structure of the convolutional layers might improve our results, as well as an investigation on what made the pooling layers prevent the learning process. Residual connections could also be implemented to help the decoder reconstructing the input. A real-time implementation could also be considered.

Audio samples for this work can be found on GitHub: github.com/Rickr922/dist-nNet.

References

1. Idmt dataset. www.idmt.fraunhofer.de/en/business_units/m2d/smt/audio_effects.html
2. Covert, J., Livingston, D.: A vacuum-tube guitar amplifier model using a recurrent neural network. In: Proceedings of IEEE SOUTHEASTCON 2013, pp. 1–5 (2013)
3. Damskägg, E.P., Juvela, L., Välimäki, V.: Real-time modeling of audio distortion circuits with deep learning. In: Proceedings of International Sound and Music Computing Conference (SMC), Malaga, Spain, pp. 332–339 (2019)
4. Damskägg, E.P., Juvela, L., Thuillier, E., Välimäki, V.: Deep learning for tube amplifier emulation. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, pp. 471–475 (2019)
5. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016). http://www.deeplearningbook.org

6. Hinton, G.E., Zemel, R.S.: Autoencoders, minimum description length and Helmholtz free energy. In: Proceedings of the 6th International Conference on Neural Information Processing Systems, NIPS 1993, pp. 3–10. Morgan Kaufmann Publishers Inc., San Francisco (1993)
7. Lu, X., Tsao, Y., Matsuda, S., Hori, C.: Speech enhancement based on deep denoising auto-encoder. In: Proceedings of Interspeech 2013, pp. 436–440 (2013)
8. Martínez Ramírez, M.A., Benetos, E., Reiss, J.D.: A general-purpose deep learning approach to model time-varying audio effects. In: 22nd International Conference on Digital Audio Effects (DAFx-19) (2019)
9. Martínez Ramírez, M.A., Reiss, J.: End-to-end equalization with convolutional neural networks. In: Proceedings of International Conference on Digital Audio Effects (DAFx) 2018, Aveiro, Portugal, pp. 296–303 (2018)
10. Martínez Ramírez, M.A., Reiss, J.D.: Modeling nonlinear audio effects with end-to-end deep neural networks. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, pp. 171–175 (2019)
11. Moreno, J.A., Bigoni, F., Palamas, G.: Latent birds: a bird's-eye view exploration of the latent space. In: Proceedings of 17th Sound and Music Computing Conference, Torino, 24th–26th June 2020 (2020)
12. van den Oord, A., et al.: WaveNet: a generative model for raw audio (2016). https://arxiv.org/abs/1609.03499. Accessed 01 Nov 2019
13. Rethage, D., Pons, J., Serra, X.: A wavenet for speech denoising. In: Proceedings of 2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, pp. 5069–5073 (2018)
14. Schattschneider, J., Olzer, U.: Discrete-time models for nonlinear audio systems. In: Proceedings of the 2nd COST G-6 Workshop on Digital Audio Effects (DAFx99) (1999)
15. Schmitz, T., Embrechts, J.J.: Real time emulation of parametric guitar tube amplifier with long short term memory neural network. In: Proceedings of Conference on Image Processing and Pattern Recognition (IPPR 2018), pp. 149–157 (2018)
16. Zhang, Z., Olbrych, E., Bruchalski, J., McCormick, T.J., Livingston, D.L.: A vacuum-tube guitar amplifier model using long/short-term memory networks. In: Proceedings of IEEE SOUTHEASTCON 2018, pp. 1–5 (2018)